# From Functional Data to Smooth Functions

Pang Du

Department of Statistics
Virginia Tech

# Overview

- We need a flexible method for constructing functions from noisy discrete data.
- The method should be able to reproduce any feature that interests us in a function, no matter how complicated.
- The computation should be reasonably fast, even when tens or hundreds of thousands of discrete values are available.
- We first consider the most popular technique, *basis function expansions*.
- We then introduce the more appealing technique, *smoothing by roughness penalty*.

# Overview

- We describe two basis function systems in detail:
  - Fourier bases
  - B spline bases
- as well as some other important systems.
- We also ask about how to estimate derivatives.

# Basis representation

- A basis function system is a set of $K$ known functions $\phi_k(t)$ that are:
    - linearly independent of each other,
    - can be extended to include any number K in the system.
- A function $x(t)$ is constructed as a linear combination of these basis functions:

$$x(t) = \sum_{k=1}^{K} c_k \phi_k(t).$$

- If vector $\mathbf{c}$ contains the coefficients, and the vector $\phi$ contains the basis functions, then

$$x(t) = \mathbf{c}' \phi(t).$$

# Derivatives in basis function systems

- In principle, computing derivatives is easy:

$$D^m x(t) = \sum_{k=1}^{K} c_k D^m \phi_k(t).$$

- but not all basis functions have derivatives that behave reasonably, or can even be calculated.

# The monomial basis

- Polynomials are perhaps the oldest and best known basis function expansion.
- A polynomial is the form:

$$x(t) = \sum_{k=1}^{K} c_k t^{k-1}.$$

- The basis functions are the monomials: $1, t, t^2, t^3, \ldots$
- Polynomials can work fine for simple problems only requiring $K = 5$ or so, but have severe problems tracking sharp localized features, and can run into computational problems for unequally spaced data.

# Derivatives of polynomials

- Derivative estimation is a big problem for polynomials because their derivatives become less and less complex, the higher the order of derivative.
- For a polynomial of degree m, the derivative of order $m + 1$ is zero.
- But in most real world systems, derivatives become more complex as the order of the derivative increases.

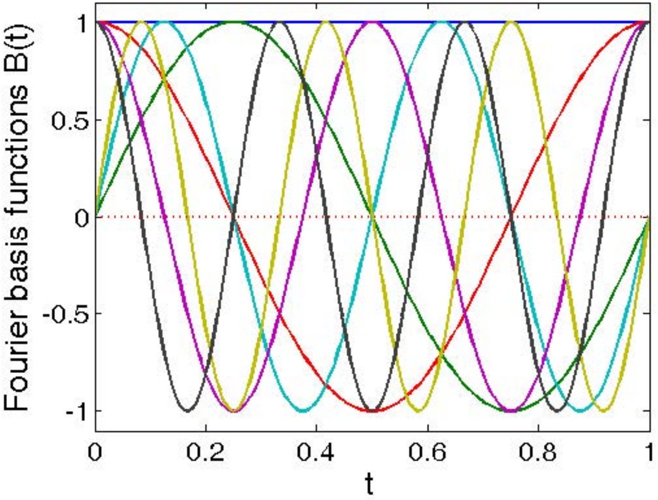
# Fourier basis functions

- The basis functions are sine and cosine functions of increasing frequency:

$$1, \sin(\omega t), \cos(\omega t), \ldots, \sin(m\omega t), \cos(m\omega t).$$

- The constant $\omega$ defines the period of oscillation of the first sine/cosine pair. That is, $\omega = 2\pi/P$ where $P$ is the period.
- $K = 2M + 1$ where $M$ is the largest number of oscillations in period P that are required.

# Fourier basis functions



# Advantages of Fourier basis functions

- Fourier bases were the only alternative to monomial bases until the middle of the 20th century.
- They have excellent computational properties, especially if the times of observation are equally spaced.
- They are natural for describing data which are periodic, such as the annual weather data, gait cycle data and so on.
- Their periodicity is a problem, however, for nonperiodic data, such as the growth curves.
- But the Fourier basis is still the first choice in many fields, such as signal analysis, even when the data are not periodic.

# Derivatives of Fourier bases

- Computing derivatives is easy since

$$D\sin(\omega t) = \omega\cos(\omega t), D\cos(\omega t) = -\omega\sin(\omega t).$$

- We say that this system is closed under differentiation: the derivative of a Fourier series expansion is also a Fourier series expansion.
- The Fourier series is infinitely differentiable.

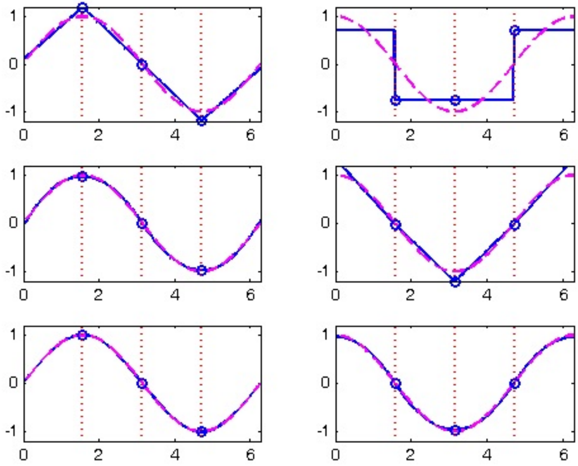
# The splines

- Splines are polynomial segments joined end-to-end.
- The segments are constrained to be smooth at the join. The values of $t$ at which adjacent segments are joined are called *knots*.
- The order $m$(order $=$ degree $+\, 1$) of the polynomial segments and the location of the knots define the spline basis system.

# An example of spline functions

- The following figure shows splines of three orders, each with three knot values.
- The splines are defined so as to offer the best fit to a sine function, shown in the left panels.
- How well the derivatives of these splines fit the derivative of the sine, the cosine, is shown in the right panels.

# An example of spline functions



# Derivatives with splines

- Because splines are constructed from polynomials, computing their derivative at any point between two knots is simple. There, the highest nontrivial order of derivative is $m-1$ for order m splines.
- At a knot, it is usual to require that the derivatives up to order $m-2$ also join. That is, the derivative of order $m-2$ of a spline function is usually continuous.
- The most popular choice of order is 4, implying continuous second derivatives. The second derivatives have straight line segments.

## Spline functions and DFs

- How can we quantify the flexibility of a spline function of order $m$?
- In the usual case, there are $m-1$ constraints on the adjacent polynomials, corresponding to the requirement that $m-2$ derivatives plus the function values are required to match at the knot.
- Given the first segment, with $m$ degrees of freedom, this means that we gain one degree of freedom with each knot to the right of the first segment.
- The total number of degrees of freedom is

$$\text{order } m + \text{number of interior knots.}$$

# How are knots chosen?

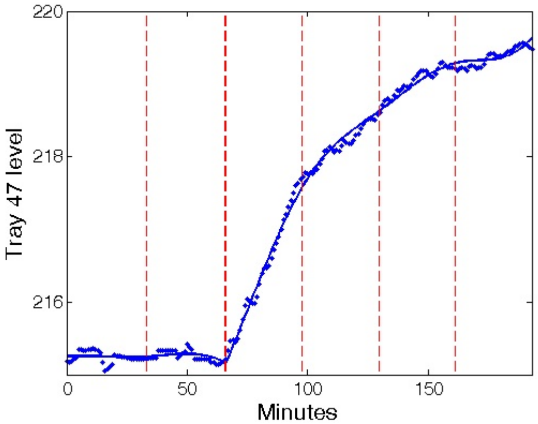
- Knots are often spaced equally.
- But two important rules should be observed in placing knots:
  - Place more knots where you know there is strong curvature, and fewer where the function changes slowly.
  - But be sure that there is at least one data point in any interval.
- Some splines, such as smoothing splines, place a knot at each point of observation.

# Coincident knots in splines

- Sometimes we need less smoothness at a specific point.
- For example, we will see problems where a function needs to be continuous at a point, but its derivative is discontinuous.
- When multiple knots are placed at the same point, the convention is that a spline loses one derivative for each additional knot.
- An order 4 spline with 3 coincident knots is continuous at that point, but does not have a first derivative.
- An order 4 spline with 4 coincident knots is discontinuous at that point.

# An example: refinery data

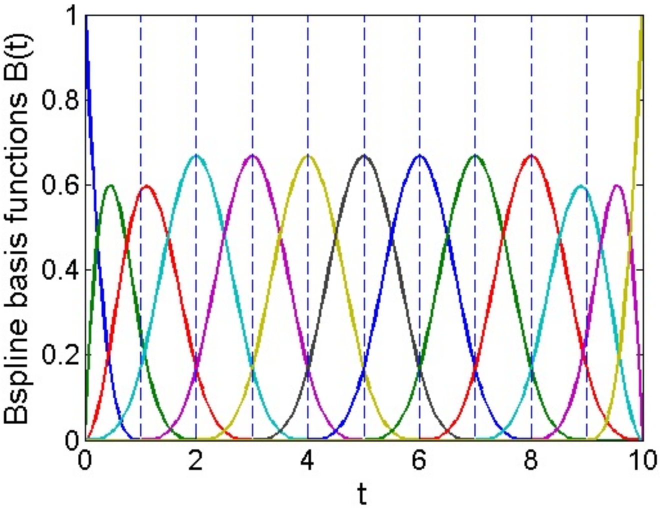
There are three coincident knots at the second location for the refinery data to permit a discontinuous first derivative.



# The B-spline basis system

- Any spline function with $K$ degrees of freedom can be expressed as a linear combination of $K$ basis spline functions.
- Among many possibilities, the B-spline system, developed in the 1940s, is the most popular.
- B-spline basis functions are themselves spline functions.
- Any B-spline basis function is positive over at most $m$ adjacent intervals.
- This ensures that computation is fast for even tens of thousands of basis functions.

# 13 order 4 B-spline basis functions



# Power basis and exponential basis systems

- **Power basis**: $t^{\lambda_1}, t^{\lambda_2}, \ldots$, where the powers are distinct but not necessarily integers or even positive.
- **Exponential basis**: $e^{\lambda_1 t}, e^{\lambda_2 t}, \ldots$, where the $\lambda$'s are distinct.

# Wavelet basis functions

- A recent development, wavelet bases combine some of the advantages of both Fourier and B-spline bases.
- They are especially good at tracking sharp highly localized features,
- and separating a signal into components which reflect both specific frequencies and specific locations on the $t$-axis. Because of their computational efficiency, they are often used for image compression.
- For example, the FBI uses wavelets to store fingerprint information.

# Empirical basis functions

- We will look at *functional principal components analysis* later.
- This is essentially a method for estimating orthogonal basis functions from functional data that capture as much of the variation as possible given a fixed number of basis functions $K$.

# Where do we go from here?

- Now we need to see how to fit a basis function expansion to noisy data.
- The simplest process is through least squares approximation (including kernel regression and local polynomials).
- This is essentially the use of multiple regression analysis, where the covariates are the basis function values corresponding to time sampling points.
- This works reasonably well, but we will see how to do even better later.

# Why roughness penalties?

- Controlling smoothness by limiting the number of basis functions is discontinuous; roughness penalties allow continuous control over smoothness.
- We want to be able to define "smooth" in ways that are appropriate to our problems.
  - We may want a smooth derivative rather than just a smooth function.
  - What is smooth in one situation is not smooth in another. Smoothness has to be defined differently for periodic functions, for example.
- We find that roughness penalty smoothing gives better results.
- Roughness penalties are connected to fitting data by a differential equation; they are models for process dynamics.

# Objectives

We have two competing objectives:

- Fit the data well; keep bias low.
- Keep the fit smooth so as to
  - filter out noise
  - get better estimates of derivatives

$$\text{Mean squared error} = \text{Bias}^2 + \text{Sampling Variance}.$$

We can often greatly reduce MSE by trading a little bias off against a lot of sampling variance.

# Quantifying roughness

- The classic: **curvature in the function**

$$\mathrm{PEN}_2(x) = \int [D^2 x(s)]^2 ds.$$

  $[D^2x(s)]^2$ measures the squared curvature in $x$ at $s$. This penalty measures *total squared curvature*.

- **Curvature in acceleration**:

$$\mathrm{PEN}_4(x) = \int [D^4 x(s)]^2 ds.$$

- These two penalties also define what we mean by "smooth"; any function that has zero penalty is "hyper-smooth". A straight line for the classic, a cubic polynomial for the acceleration penalty.

# Harmonic acceleration

- If the process is periodic, it is natural to think of a constant + sinusoid as "hyper-smooth". This suggests that we use

$$PEN_H(x) = \int [D^3x(s) + \omega^2 Dx(s)]^2 ds,$$

  where $2\pi/\omega$ is the period.
- The functions $1, \sin(\omega t)$, and $\cos(\omega t)$ all have zero penalties, as does any linear combination of them.

# Some questions to think about

- Writing $Lx(s) = D^3x(s) + \omega^2 Dx(s)$, we have
$$PEN_H(x) = \int [Lx(s)]^2 ds.$$
- Can we think of other differential operators $L$ that might be useful?
- If we have a small number of "hyper-smooth" functions in mind, can we find a differential operator $L$ that will assign zero penalty to them?
- Can we use the data themselves to tell us something about the right differential operator $L$?

# Notation

- Notation:
  - $\mathbf{y}$ is the $n$-vector of data $y_j$ to be smoothed.
  - $\mathbf{t}$ is the $n$-vector of values of $t_j$.
  - $\mathbf{W}$ is a symmetric positive definite weight matrix, allowing for possible covariance structure among residuals.
  - $x(\mathbf{t})$ is the $n$-vector of fitted values, and $x(\mathbf{t})$ has the basis function expansion

$$x(\mathbf{t}) = \sum_{k=1}^{K} c_k \phi_k(\mathbf{t}) = \mathbf{c}'\phi(\mathbf{t}).$$

- The penalized least squares criterion is

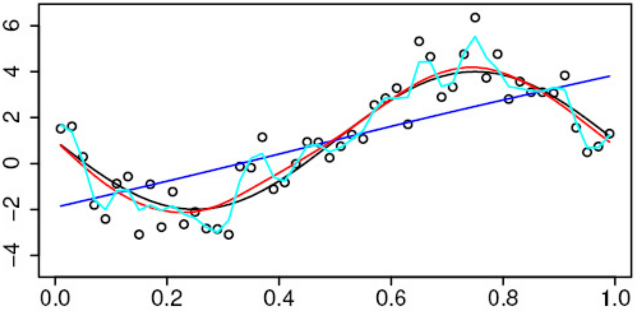
$$\mathrm{PENSSE}_\lambda(x|\mathbf{y}) = [\mathbf{y} - x(\mathbf{t})]'\mathbf{W}[\mathbf{y} - x(\mathbf{t})] + \lambda \mathrm{PEN}(x)$$

# How the smoothing parameter works?

*Smoothing parameter* $\lambda$ controls the amount of roughness.

- As $\lambda \to 0$, roughness matters less and less, and $x(\mathbf{t})$ fits the data better and better.
- As $\lambda \to \infty$, roughness matters more and more, and $x(\mathbf{t})$ becomes more and more "hyper-smooth".
- Our job is to find the right value where we trade enough bias off against sampling variance to minimize mean squared error.

# Effect of smoothing parameter




Three estimates of $x$ are calculated by minimizing $\sum_{i=1}^{n}(y_i - x(t_i))^2 + n\lambda \int_0^1 [x''(t)]^2 dt$ at $\log_{10} n\lambda = 0, -3, -6$.

# The roughness penalty matrix

- For the classic penalty,

$$\begin{aligned}\mathrm{PEN}_2(x) &= \int [D^2 \mathbf{c}'\phi(t)]^2 dt \\ &= \mathbf{c}' \int [D^2\phi(t)][D^2\phi(t)]' dt \mathbf{c} \\ &= \mathbf{c}'\mathbf{R}\mathbf{c}.\end{aligned}$$

- The order-$m$ roughness penalty matrix $\mathbf{R}$ is

$$\mathbf{R} = \int [D^m\phi(t)][D^m\phi(t)]' dt = \int (D^m\phi)(D^m\phi)'.$$

- substitute $L$ for $D^m$ for more general roughness penalties.

# The estimates for $\mathbf{c}$ and $\mathbf{y}$

- $\Phi$ is the $n \times K$ matrix of basis function values $\phi_k(t_j)$.
- The penalized least squares criterion becomes

$$\mathrm{PENSSE}(\mathbf{y}|\mathbf{c}) = (\mathbf{y} - \Phi\mathbf{c})'\mathbf{W}(\mathbf{y} - \Phi\mathbf{c}) + \lambda\mathbf{c}'\mathbf{R}\mathbf{c}.$$

- This is quadratic in $\mathbf{c}$, and is minimized by

$$\hat{\mathbf{c}} = (\Phi'\mathbf{W}\Phi + \lambda\mathbf{R})^{-1}\Phi'\mathbf{W}\mathbf{y}.$$

# The smoothing matrix $\mathbf{S}_{\phi,\lambda}$

- The data-fitting vector $\hat{\mathbf{y}} = x(\mathbf{t})$ is

$$\hat{\mathbf{y}} = \Phi(\Phi'\mathbf{W}\Phi + \lambda\mathbf{R})^{-1}\Phi'\mathbf{W}\mathbf{y}.$$

- Smoothing matrix

$$\mathbf{S}_{\phi,\lambda} = \Phi(\Phi'\mathbf{W}\Phi + \lambda\mathbf{R})^{-1}\Phi'\mathbf{W}$$

maps the data into the fit, and has many useful applications.

# Equivalent degrees of freedom

- It is useful to compare a fit using a roughness penalty to one using a fixed number of basis functions.
- A measure of the "degrees of freedom" in a roughness penalized fit is

$$df(\lambda) = \text{trace}(\mathbf{S}_{\phi,\lambda}).$$

- This corresponds to the number of basis functions $K$ in an un-penalized fit.

# Cross-validation

If we choose smoothing parameter $\lambda$ by cross-validation, we

- set aside a subset of data, the **validation sample**
- call the remaining data the **training sample**
- fit the model to the training sample
- assess fit to the validation sample
- repeat these steps a number of times (often five or ten times), and choose the $\lambda$ value that gives the best fit averaged across the repetitions.

# Delete-one cross-validation

We can also, for a sequence of values of $\lambda$,

- set aside each observation $(t_j, y_j)$ in turn
- fit the data with the rest of the sample,
- sum fits to the left out values to get a *cross-validated error sum of squares* $\mathrm{CV}(\lambda)$.
- select the $\lambda$ value that minimizes $\mathrm{CV}(\lambda)$.

# Generalized cross-validation

- Cross-validation is time-consuming, and tends too often to under-smooth the data.
- The generalized cross-validation criterion is

$$\mathrm{GCV}(\lambda) = \left(\frac{n}{n - df(\lambda)}\right)\left(\frac{\mathrm{SSE}}{n - df(\lambda)}\right),$$

  where $df$ is the equivalent degrees of freedom of the smoothing operator.
- $\mathrm{GCV}(\lambda)$ approximates $\mathrm{CV}(\lambda)$.
- The right factor is just the unbiassed estimate $\sigma_e^2$ of residual variance familiar in regression analysis.
- The left factor "discounts" this measure further to allow for the influence of optimizing with respect to $\lambda$.

# Simulation: Jolicoeur's growth model

- How does GCV work in a simulated data example?
- A parametric growth model by Pierre Jolicoeur at the Université de Montréal (Canada) offers a nice test problem.
- The growth curve in Jolicoeur (1992) has the form

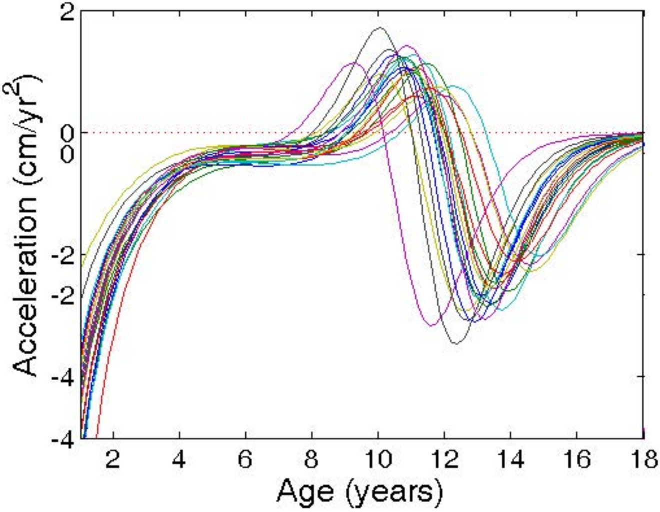
$$h(t) = \frac{a \sum_{k=1}^{3} [b_k(t+e)]^{c_k}}{1 + \sum_{k=1}^{3} [b_k(t+e)]^{c_k}}.$$

- How well do we estimate the Jolicoeur acceleration curves?

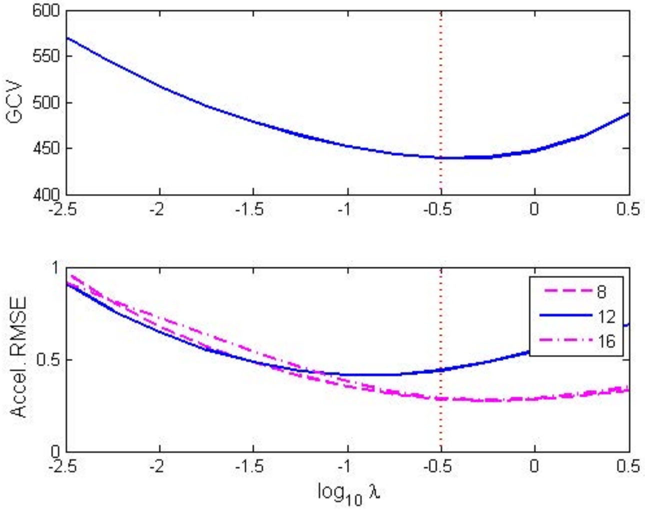
# Simulation: Setup

- Parameter fits to the Fels growth data (Roche, 1992) by Bock (2000): $a = 164.7, e = 1.474, \mathbf{b} = (0.3071, 0.1106, 0.0816)', \mathbf{c} = (3.683, 16.665, 1.474)'$.
- Standard error curve $s(t)$ is also available in the textbook, and used to simulate normal random errors that are added to the smooth growth curve.
- Sampling ages $t_i$: quarterly for 1-2 years, annually for 2-8 years, and every other year after that to 18 years of age.
- $1/s^2(t_i)$ define the entries of the diagonal weight matrix $\mathbf{W}$.
- We simulate 1000 samples.
- We smooth using a range of values of $\lambda$, and note the value giving the best value of GCV.

# 20 Jolicoeur acceleration curves



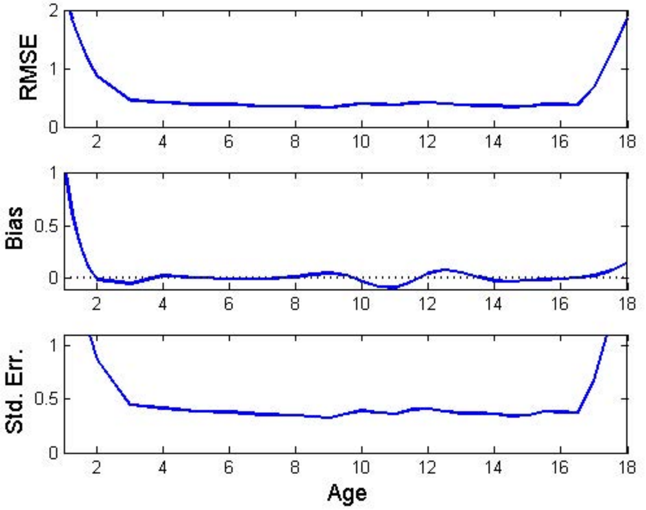
# GCV and RMSE


600
550
500
450
400
GCV
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
0.5
0
Accel. RMSE
8
12
16
$\log_{10} \lambda$

# What we see?

- In the top panel, GCV favors $\lambda = 0.1$.
- This is about right for optimal RMSE for ages 8 and 16, but less smoothing would be better for age 12, in the middle of the pubertal growth spurt.
- One smoothing parameter value does not work best for all ages, but
- The value chosen by GCV certainly does a fine job.

# RMSE, bias, and SE



# What we see?

- The performance of the smoothing spline estimate deteriorates badly at the extremes.
- The sharp curvature at the pubertal growth spurt also causes some problems.
- Except at the extremes and PGS, the bias is negligible. The standard error is about the same as RMSE.
- Would we do better at the extremes if the smooth respected monotonicity?

# Summary

- Roughness penalization, also called *regularization*, is a flexible and effective way to ensure that an estimated function is “smooth”.
- We can tailor the definition of “smooth” to our needs.
- The roughness penalty idea extends to any type of *functional parameter* that we want to estimate from the data.
- Roughness penalties are one of the main ways in which we exploit the smoothness that we assume in the process generating the data.

# Roughness and energy

- "Roughness" is like *energy* in physics
- Roughness requires energy to produce, and smoothness implies limited energy.
- Where we imagine that the amount of energy behind the data is limited, it is natural to assume smoothness.